LuxSenz transmitter software

The LuxSenz receiver software runs on the STM32F446RE microcontroller that is present on a Nucleo-F446RE board in the demo transmitter. It accepts input from a USB keyboard and some sensors, and generates a control signal for the transmitting shutters.

The source files are made to give a demo of the LuxSenz technology. The Nucleo-F446RE was selected because it was already available and it supports full-speed USB. The microcontroller and the Nucleo board are not optimized for low-power applications. We recommend to adapt the code for a low-power microcontroller (removing USB support) if you want to use LuxSenz for transmitting data in a low-power system.

# Building the software

The software was developed using Visual Studio and the VisualGDB plugin. Another IDE could be used too, as long as the used compiler (GCC) is the same.

# Loading the software on the microcontroller

Disconnect all power inputs from the Nucleo board. Set jumper JP5 to U5V and make sure that the jumpers on CN2 are placed. Connect the board to a computer using a mini-USB cable. Use VisualGDB or STLINK utility to load the program on the microcontroller.

# Inputs

The software supports Si7021 and SGP30 sensor boards (and has partial support for MPL115A2 boards), to transmit data about TVOC gas concentration, CO2 concentration, temperature, humidity and (possibly) air pressure. Sensors are polled frequently as required for some of the sensors and data will be automatically transmitted once in a while.

A USB keyboard can be connected to send other messages. By typing a message and pressing the return key (enter), the typed-in message will be transmitted. Furthermore, the following shortcuts exists:

* Control + Backspace Send reset command (will refresh e-ink display on the receiver)
* Enter Send/Resend last typed message.
* Control + D Shortcut to type a DC2 (device control 2) character. This character is the first of any control message that can be found in the LuxSenz communication protocol.

# Source files

|  |  |  |
| --- | --- | --- |
| Stm32f4xx\_hal\_\*.c | | Hardware Abstraction Layer libraries for the STM32 microcontroller |
| Stm32f4xx\_hal\_conf.h | | Header file to enable/disable certain modules of the HAL library |
| Startup\_stm32f446xx.c  System\_stm32f4xx.c | | Default files to let the STM32F446 start properly, made by the MCD Application Team |
| Stm32f4xx\_nucleo\_144.c | | Firmware for a Nucleo144 board (different board, with a USB connector by default) with an STM32F4xx microcontroller. |
| Stm32f4xx\_ll\_gpio.h  Stm32f4xx\_ll\_\*.c | | Files from the Low-Layer library |
| Sensirion\_common.h  Sensirion\_common.c  Sensirion\_configuration.h  Sensirion\_configuration.c  Sgp\_featureset.h  Sgp30\_featureset.c  Si7021\_driver.h  Si7021\_driver.c | i2c.h  MPL115A2.h  MPL115A2.c  Sgp30.h  Sgp30.c | Code to readout the sensors using i2c, collected from several other repositories. |
| Usbh\_conf.h | Usbh\_conf.c | USB initialization code |
|  | |  |
| Stm32f4xx\_it.h | | Header file of all interrupt service routines |
| Main.h | | Header file of the main source file. In this file one can:   * Set the FSK frequencies * Set the duty cycle for FSK frequency 1 * Set a small change to the duty cycle for FSK frequency 2, to balance the average shutter opacity between both frequencies (should be the same to avoid flicker) |
| Stm32f44xx\_it.c | | * Interrupt handlers to generate the control signal for the shutters that transmits data * Functions that can be called to set the message that will be transmitted |
| Stm32f4xx\_hal\_msp.c | | GPIO Initialization code that is called by the HAL library |
| Menu.c  Mouse.c | | Files from the USB example code, that is largely not used, but is still there in the repository |
| Keyboard.c | | Code to process keyboard input. |
| Main.c | | * Initialization code * Main program body, that reads data from the sensors * Random adjustments to previously read sensor data, when a sensor cannot be read properly * Replacement function for HAL\_Delay that allows USB input during timed delays |

# Code structure

The output signals are generated using a timer interrupt, that is set depending on the output signal that needs to be generated (timer interrupt value is updated after every call). Sensors and USB input is handled in the main program body loop.